1. OpenCV를 사용하여 이미지를 불러오고 화면에 출력

• 원본 이미지와 그레이 스케일로 변환된 이미지를 나란히 표시

• 요구사항

• cv.imread()를 사용하여 이미지 로드

• cv.cvtColor() 함수를 사용해 이미지를 그레이 스케일로 변환

• np.hstack() 함수를 이용해 원본이미지와 그레이 스케일 이미지를 가로로 연결하여 출력

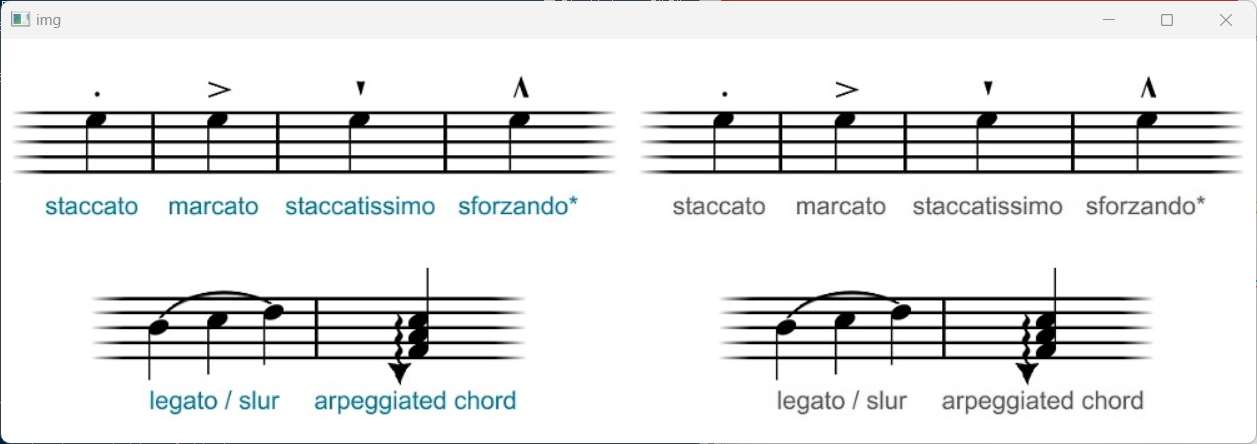
• cv.imshow()와 cv.waitKey()를 사용해 결과를 화면에 표시하고, 아무키나 누르면 창이 닫히도록 할것

• 힌트:

• OpenCV는이미지를 BGR 형식으로읽음

• 그레이 스케일 변환시 cv.COLOR\_BGR2GRAY 사용

풀이 : 이미지를 이어 붙이는 코드는 이미 6번 PPT에 np.hstack()을 통해 이미지를 합치는 코드의 완성본이 있으므로 그 코드를 가져오는것으로 요구사항을 대부분 충족하지만, 이 문제의 핵심은 그레이 스케일 변환된 이미지는 RGB 데이터가 없으므로 컬러 이미지와 흑백 이미지의 마지막 (세번째) 차원이 달라 그냥 붙일 수 없는 문제를 해결하는 것이 관건이다. 본인은 np.stack()을 통해 흑백 이미지의 밝기 값을 각 B,G,R값에 할당된 tmp 이미지를 생성해서 이것을 컬러 이미지와 합치는것으로 해결했다. 실행할 때 이미지는 같은 폴더내의 a.jpg를 읽음  
코드는 1.이미지 불러오기 및 그레이스케일 변환.py이다.



결과물

2.웹캠 영상에서 에지 검출

• 설명

웹캠을 사용하여 실시간 비디오 스트림을 가져온다.

각 프레임에서 Canny Edge Detection을 적용하여 에지를 검출하고 원본 영상과 함께 출력한다.

• 요구사항

cv.VideoCapture()를 사용해 웹캠 영상을 로드한다.

각 프레임을 그레이스케일로 변환한 후, cv.Canny() 함수를 사용해 에지 검출을 수행한다.

원본 영상과 에지 검출 영상을 가로로 연결하여 화면에 출력한다.

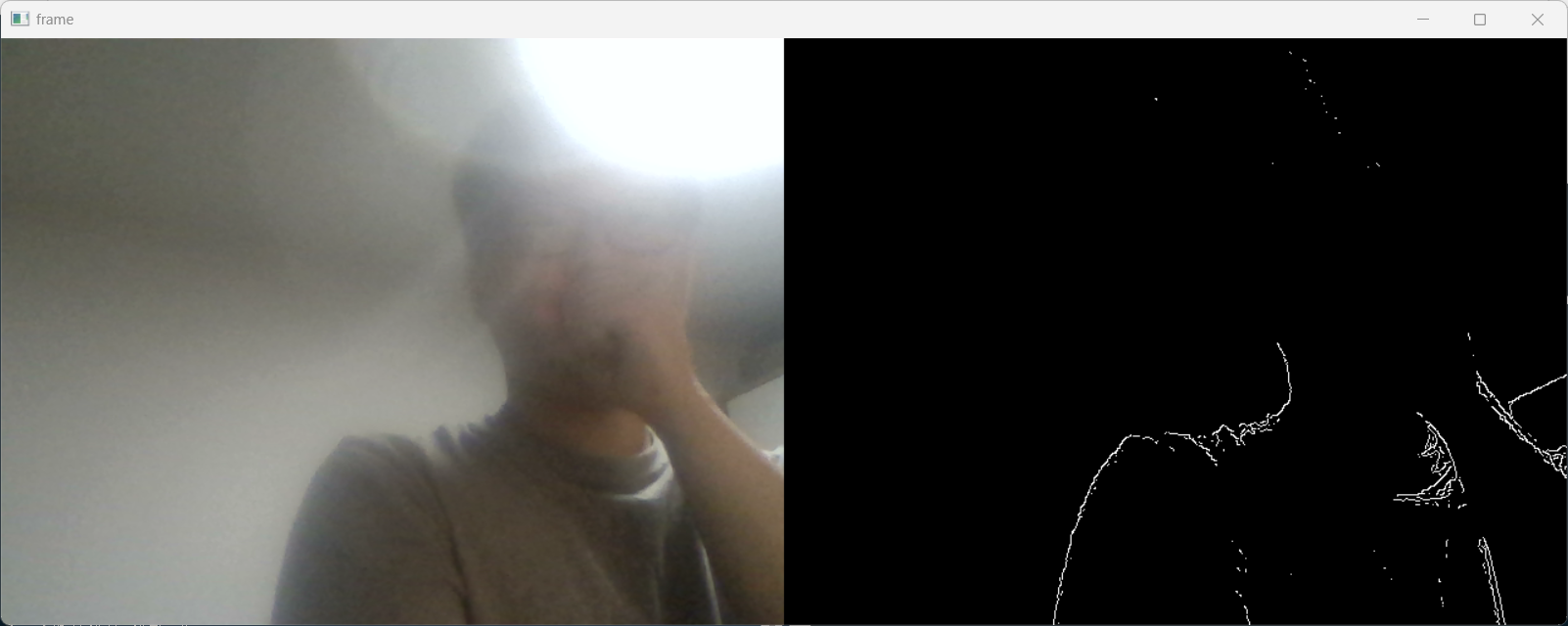
q 키를 누르면 영상 창이 종료된다.

• 힌트

cv.Canny() 함수는 에지 검출에 사용되며, 하한/상한 임계값을 인자로 받는다.

cv.destroyAllWindows()를 사용해 창을 닫는다.

풀이 : 과제 1번과 수업 자료 5번을 합쳐서 만드는 과제, 웹 캠에서 이미지를 읽는 코드는 수업 자료 5번에 있고 (cv.VideoCapture 및 cap.read()함수 사용) 과제 1에서 쓴 np.stack() 함수를 사용해서 에지를 추출한 흑백 이미지를 합친다. 에지를 추출하는 파라미터는 요구사항이 없으므로 첫번째 인자가 이미지라는것만 안다. (실제 코드에선 cv.Canny(gray\_img, 50, 50) 으로 사용) 따라서 과제 1번을 푼 상태에서 수업 자료 5번에서 나오는 각각의 코드가 무슨 역할을 알고 순서대로 배열 할 수 있다면 쉽게 풀 수 있다. (5번의 14번 라인 밑에 cv.imshow()함수 사이에 1번 코드의 cv.cvtColor(frame,cv.COLOR\_BGR2GRAY)를 사용한 흑백 변환 및 에지 추출 코드와 과제 1번에서 한 차원 확장을 해주면 된다. )

결과물:

OpenCV 03: 마우스로 영역 선택 및 ROI(관심 영역) 추출

• 설명

이미지를 불러오고 사용자가 마우스로 클릭하고 드래그하여 관심 영역(ROI)을 선택한다.

선택한 영역만 따로 저장하거나 표시한다.

• 요구사항

이미지를 불러오고 화면에 출력한다.

cv.setMouseCallback()을 사용하여 마우스 이벤트를 처리한다.

사용자가 클릭한 시작점에서 드래그하여 사각형을 그리며 영역을 선택한다.

마우스를 놓으면 해당 영역을 잘라내서 별도의 창에 출력한다.

r 키를 누르면 영역 선택을 리셋하고 처음부터 다시 선택할 수 있다.

s 키를 누르면 선택한 영역을 이미지 파일로 저장한다.

• 힌트

cv.rectangle() 함수를 사용하여 드래그 중인 영역을 시각화한다.

ROI 추출은 NumPy 슬라이싱을 사용한다.

cv.imwrite()를 사용하여 이미지를 저장한다.  
  
수업 자료의 8번 및 9번 코드를 통해서 콜백함수를 통해 실시간으로 사각형이 그려지는건 구현할 수 있다.

다만 이를 다시 지우는 코드는 (지우지 않으면 사각형이 끊임없이 중첩되어 그려진다.) 수업 자료나 힌트에 적혀있지 않았는데 이를 수행할 수 있는 방법을 여러 방면으로 찾아보니 원본 이미지를 다시 불러오고 imshow()함수로 이미지를 다시 로딩하는 것이 가장 빠른 방법이라고 결론을 내렸다. 따라서 다음과 같이 draw 함수를 구현했다.

def draw(event, x, y, flags, param):

global ix, iy, img, img\_tmp, ismoved

if event==cv.EVENT\_LBUTTONDOWN:

ix, iy=x, y

ismoved=False

elif event==cv.EVENT\_MOUSEMOVE and flags == cv.EVENT\_FLAG\_LBUTTON:

ismoved=True

img = img\_tmp.copy()

cv.rectangle(img,(ix, iy), (x, y), (255,0,0), 2)

cv.imshow('ROI', img)

elif event==cv.EVENT\_LBUTTONUP:

if(ismoved):

img = img\_tmp[min(iy, y):max(iy, y), min(ix, x):max(ix, x)]

img\_tmp= img.copy()

cv.imshow('ROI', img)

여기서 완전 원본 이미지를 img\_ori (이는 초기에 수행된다.), 사각형이 그려지기 전 img는 img\_tmp, 실제로 도형이 그려지는 img는 img로 저장한다. (img.copy()함수 사용으로 복사)

또한 ROI를 슬라이싱 하는 것은 img 변수의 슬라이싱([:](을 사용하되 opencv 이미지의 차원 표현img[y][x]꼴임에 유의하고 은 사각형이 아래에서 위(혹은 오른쪽에서 왼쪽)으로 그려지는 경우도 정상적으로 잘릴 수 있게 iy, y / ix, x (버튼 눌렀을떄 시작 지점, 땠을때 지점)을 min, max를 통해서 구현했다.

또한 r키를 눌렀을 때 img 변수에 img\_ori()내용을 복사하고 cv.imshow()함수를 사용해서 새로고침함

저장은 요구한대로 cv.imwrite(경로, 이미지)를 사용해서 저장해준다.

